LABORATORY FOR COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MIT/LCS/TM-431

THE EMERGING THEORY OF AVERAGE-CASE COMPLEXITY

Robert E. Schapire

June 1990

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) N00014-89-J-1988
6a. NAME OF PERFORMING ORGANIZATION MIT Lab for Computer Science 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Dept. of Navy
6c. ADDRESS (City, State, and ZIP Code) 545 Technology Square Cambridge, MA 02139	7b. ADDRESS (City, State, and ZIP Code) Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

The Emerging Theory of Average-case Complexity.

12. PERSONAL AUTHOR(S)
Robert E. Schapire

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	6/90	17

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Complexity, average-case complexity.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Abstract

This paper reviews some of the recent results that have emerged in the study of average-case complexity. Included is a description of Levin's framework for studying average-case complexity, as well as his proof of the existence of "complete" problems for a class of distributional problems. The paper also presents some new results, including a natural and more liberal extension of Levin's model, in addition to a partial characterization of the relationships among the new average-case complexity classes.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Judy Little	22b. TELEPHONE (Include Area Code) (617) 253-5894 22c. OFFICE SYMBOL

The Emerging Theory of Average-case Complexity

Robert E. Schapire

MIT Laboratory for Computer Science
Cambridge, MA 02139

June 11, 1990

Abstract

This paper reviews some of the recent results that have emerged in the study of average-case complexity. Included is a description of Levin's framework for studying average-case complexity, as well as his proof of the existence of "complete" problems for a class of distributional problems. The paper also presents some new results, including a natural and more liberal extension of Levin's model, in addition to a partial characterization of the relationships among the new average-case complexity classes.

Keywords: Complexity, average-case complexity.

1 Introduction

A primary contribution of theoretical computer science has been the identification of the so-called NP-complete problems, a well-known class of problems provably equivalent to one another in worst-case computational complexity, modulo polynomial-time computation. These problems, being the "hardest" in the class NP, are widely believed to be unsolvable by any polynomial-time algorithm, and indeed, no sub-exponential time algorithm is known for any NP-complete problem.

Nevertheless, at least since the late 1970's, algorithms have been known that can solve assorted NP-complete problems in polynomial-time *in the average case*, i.e., whose expected running time on an instance chosen randomly (according to some "natural" distribution) is bounded by a polynomial. Johnson [13] surveys a number of these results, including, for instance, expected polynomial-time algorithms for finding Hamiltonian circuits in random graphs, and for 3-coloring random graphs. Typically, such algorithms are based on the observation that almost all random instances have some easily observed property that makes the decision problem trivial; the remaining few instances can then be solved by an exponential-time, brute-force algorithm. For example, almost all random graphs contain 4-cliques which make them trivially non-3-colorable; in the extremely unlikely event that the randomly chosen graph does not contain a 4-clique, a brute-force strategy can be used to determine if the graph is 3-colorable.

Given such results, one may naturally wonder whether there exist any algorithms that are "hard" on average, and if so, how one might go about identifying such problems and proving their hardness. One approach, first suggested by Levin, is to follow the strategy set forth in the theory

This paper was prepared with support from ARO Grant DAAL03-86-K-0171, DARPA Contract N00014-89-J-1988 and a grant from the Siemens Corporation.

Author's net address: rs@theory.lcs.mit.edu.

of worst-case complexity of proving completeness for a class of problems using some appropriate notion of reducibility. Levin [14, 15] introduced his notion of average-case completeness, analogous to the usual worst-case completeness, in 1984. In his setting, problems consist of two parts: a decision problem, and a distribution on instances. The class DistNP consists of those problems whose decision problem is in NP, and whose distribution is computable in polynomial time (more details in later sections). Levin shows that a tiling problem, under an "almost" uniform distribution on the instances, is complete for DistNP. Thus, if this tiling problem is computable in polynomial-time on average, then so is every problem in the class DistNP — seemingly strong evidence that the problem is hard on average.

Levin's original paper was virtually incomprehensible in its terseness, recommended by Johnson [13] only for "cryptoanalytically inclined readers." Fortunately, Gurevich [9], Gurevich and McCauley [10] and Goldreich [5] have since provided the community the valuable service of deciphering and explaining Levin's one-and-a-half page note in expositions that far exceed Levin's both in length and clarity.

Gurevich [6, 9, 8] also managed to prove the completeness for DistNP of a few other moderately natural problems, and Venkatesan and Levin [17] were later able to find a complete graph coloring problem. Nevertheless, in general, there has been a great dearth of such results, sharply contrasting with the hundreds of natural problems known to be NP-complete [4]; apparently, proving completeness for DistNP is much harder than for NP.

In the meantime, some theoretical aspects of average-case complexity, such as the relationship of DistNP to other complexity classes, have been studied by Gurevich [6, 9] and Ben-David et al. [2]. Some of these will be described in later sections.

In this paper, I will review the development of the theory of average-case completeness outlined above. Where possible, I have also tried to make contributions to this theory. Among these contributions is an alternative characterization of "polynomial on average" that seems to simplify some of the proofs found in the literature, and that perhaps is more intuitive than the "standard" definition proposed by Levin to which it is equivalent. I also introduce in Section 4 a new and more liberal notion of "easy on average" that may be more appropriate in some settings. Finally, in Section 5, I have organized and contributed to what is known about the relationships among the various new average-case complexity classes.

2 A model for studying average-case complexity

The notation and terminology presented in this section are adopted for the most part from Goldreich [5]. We will assume for simplicity that $\Sigma = \{0,1\}$ is our input alphabet, and that Σ^*, the set of finitely long strings over Σ, is ordered in the usual lexicographic order: $0, 1, 00, 01, \ldots$. (To avoid irritating difficulties at a later point, the empty string is omitted.) We write $x < y$ if x comes before y in this ordering, and we denote by $|x|$ the length of x in symbols.

We begin with a discussion of distributions. Naturally, the average-case behavior of a program is dependent upon the distribution against which the "average" is being taken. A *density* function is a real-valued function $\mu' : \Sigma^* \to [0,1]$ mapping strings to values between 0 and 1, and for which $\sum_{x \in \Sigma^*} \mu'(x) = 1$. Thus, $\mu'(x)$ can be interpreted as the probability that x is chosen. The associated *distribution* function $\mu : \Sigma^* \to [0,1]$ is defined by

$$\mu(x) = \sum_{y \leq x} \mu'(x).$$

Clearly, μ is nondecreasing and approaches 1 asymptotically.

A *distributional problem* is a pair (D, μ) where $D : \Sigma^* \to \{0,1\}$ is a Boolean predicate, and μ is a distribution function.

Defining easy on average

As a first step to developing a theory of average-case complexity, we will need a set of careful definitions that express appropriately what is meant intuitively by a problem that is easy or hard on average.

We need first a notion of what it means for a function $f : \Sigma^* \to \mathbf{R}^+$ to grow "polynomially on average." It turns out that the most natural and intuitive definition of such a notion suffers serious deficiencies. In particular, such a definition might require that

$$\sum_{|x|=n} \mu'_n(x) \cdot f(x) \leq O(n^k) \tag{1}$$

for all n and some constant k, where $\mu'_n(x) = \mu'(x)/\sum_{|x|=n} \mu'(x)$. Thus, this definition requires that the expected value of f over inputs of length n be bounded by a polynomial in n.

Goldreich [5] and Gurevich [7] give several arguments why this is not the "right" definition. Briefly, these difficulties arise from the fact that the definition is not closed under composition with a polynomial. As a result, the definition is not machine-independent — i.e., an algorithm running in polynomial time on average (under this naive definition) on one Turing machine may no longer have this quality if the machine model is altered slightly. The definition is also dependent on the manner in which the instances are encoded; for instance, Goldreich gives an example of a graph algorithm that is fast when the input graph is encoded by its incidence matrix, but is slow when the graph is encoded by an adjacency list.

Levin [15] introduces a definition of polynomial on average that, though less intuitive in appearance, succeeds in overcoming these shortcomings. Namely, a function $f : \Sigma^* \to \mathbf{R}^+$ *is polynomial on average* with respect to distribution μ if there exists some constant $\delta > 0$ such that

$$\sum_{x \in \Sigma^*} \mu'(x) \cdot \frac{f(x)^\delta}{|x|} < \infty.$$

Here I propose an alternative, equivalent formulation of polynomial on average that may be more intuitively appealing, and that will be useful in proving some of the results that follow. This formulation also generalizes more smoothly to other notions, such as logarithmic on average, considered by Ben-David et al. [2].

A function $f : \Sigma^* \to \mathbf{R}^+$ is *usually bounded* by a function $p : \mathbf{N} \times \mathbf{R}^+ \to \mathbf{R}^+$ with respect to distribution μ if, for all $\epsilon > 0$,

$$\Pr_\mu [f(x) > p(|x|, 1/\epsilon)] < \epsilon,$$

where the probability is computed over x chosen randomly according to μ. Thus $p(\cdot, 1/\epsilon)$ bounds f for all but ϵ of the instances.

When p is restricted to be a polynomial, we obtain Levin's notion of polynomial on average.

Lemma 1 *Let $f : \Sigma^* \to \mathbf{R}^+$, and let μ be a distribution function. Then f is polynomial on average if and only if f is usually bounded by a polynomial (with respect to μ).*

Proof: Let $\delta > 0$ witness that f is polynomial on average. Then the expected value of $f(x)^\delta/|x|$ is bounded by some number N. By Markov's inequality, it follows that, for $\epsilon > 0$,

$$\Pr_\mu \left[\frac{f(x)^\delta}{|x|} > \frac{N}{\epsilon} \right] < \epsilon,$$

or equivalently,

$$\Pr_\mu \left[f(x) > \left(\frac{N|x|}{\epsilon} \right)^{1/\delta} \right] < \epsilon.$$

That is, f is usually bounded by the polynomial $(N|x|/\epsilon)^{1/\delta}$.

Conversely, suppose without loss of generality that f is usually bounded by the polynomial $(k|x|/\sqrt{\epsilon})^k$, for some constant $k > 0$. It follows that, for $\epsilon > 0$,

$$\Pr_\mu \left[\frac{f(x)^{1/k}}{|x|} > \frac{k}{\sqrt{\epsilon}} \right] < \epsilon,$$

and so, for $t > 0$,

$$\Pr_\mu \left[\frac{f(x)^{1/k}}{|x|} > t \right] < \frac{k^2}{t^2}.$$

Thus,

$$\begin{aligned} \mathrm{E}_\mu \left[\frac{f(x)^{1/k}}{|x|} \right] &\leq \sum_{t=1}^{\infty} \Pr_\mu \left[t-1 < \frac{f(x)^{1/k}}{|x|} \leq t \right] \cdot t \\ &= \sum_{t=0}^{\infty} \Pr_\mu \left[\frac{f(x)^{1/k}}{|x|} > t \right] \\ &< 1 + k^2 \cdot \sum_{t=1}^{\infty} t^{-2} < \infty, \end{aligned}$$

and therefore $1/k$ witnesses that f is polynomial on average. ■

It is now easy to see why this definition of polynomial on average is closed under composition with a polynomial: if f is usually bounded by polynomial p, then clearly f^c is usually bounded by polynomial p^c, for any constant $c > 0$.

Also, it is not hard to show that the "naive" definition of polynomial on average implies the correct definition. For suppose f satisfies Equation (1) so that, for some $k > 0$,

$$\mathrm{E}_{\mu_n} [f(x)] < kn^k.$$

Then by Markov's inequality, for $\epsilon > 0$,

$$\Pr_{\mu_n} \left[f(x) > kn^k/\epsilon \right] < \epsilon.$$

This implies that

$$\Pr_\mu \left[f(x) > k|x|^k/\epsilon \right] < \epsilon$$

and so f is usually bounded by the polynomial $k|x|^k/\epsilon$.

Thus, an algorithm A that runs in time polynomial on average can be thought of as follows: given $\epsilon > 0$ and a randomly chosen instance x, A halts in time polynomial in $|x|$ and $1/\epsilon$ with probability exceeding $1 - \epsilon$. Note again that this probability is over the random choice of x, and *not* over any kind of randomization of A. (In fact, we will usually only consider deterministic algorithms.)

We say that a distributional problem (D, μ) is *polynomial time on average* if there exists a Turing machine that decides D whose running time is polynomial on average with respect to μ.

Reducibility

We will next require a notion of reducibility. Such a notion should have the property that if (D_1, μ_1) is reducable to (D_2, μ_2), and if (D_2, μ_2) is polynomial on average, then so is (D_1, μ_1).

More formally, we say that a function $f : \Sigma^* \to \Sigma^*$ *reduces* distributional problem (D_1, μ_1) to (D_2, μ_2) if

1. f is computable in time polynomial on average (with respect to μ_1);
2. for all $x \in \Sigma^*$, $D_1(x) = D_2(f(x))$;
3. for some constant $c \geq 0$,
$$\mu_2'(x) \geq \frac{1}{|x|^c} \cdot \sum_{y \in f^{-1}(x)} \mu_1'(y).$$

The first two conditions on f are straightforward — the first requires that f be efficiently computable (on average), and the second requires that f be valid in the sense that true instances of D_1 are mapped only to true instances of D_2. The third condition is something new: here we require that common instances of D_1 not be mapped to rare instances of D_2, and that the distribution induced on D_2 by μ_1 and f not be "too far off" from μ_2.

The term "domination" is used to refer to this relationship between distributions. Thus, distribution μ_2 *dominates* μ_1 if there exists a constant $c \geq 0$ such that $\mu_2'(x) \geq |x|^{-c}\mu_1'(x)$ for all $x \in \Sigma^*$. Thus, the last condition of the above reducibility definition states that μ_2 dominates the induced distribution μ_{1f} defined by
$$\mu_{1f}'(x) = \sum_{y \in f^{-1}(x)} \mu_1'(y).$$

Finally, we are ready to prove the following theorem which justifies the preceding definitions:

Theorem 1 *Let f reduce (D_1, μ_1) to (D_2, μ_2), and suppose that (D_2, μ_2) is polynomial on average. Then so is (D_1, μ_1).*

This theorem is presented in detail by Goldreich [5] and is re-proved here as an exercise in the characterization of polynomial on average provided by Lemma 1.

Following Goldreich, we break the proof into two steps:

Step 1 *Let μ_{1f} be the distribution on instances of D_2 induced by μ_1 and f, and suppose that (D_2, μ_{1f}) is polynomial time on average. Then so is (D_1, μ_1).*

Proof: By Lemma 1, there exists an algorithm B solving (D_2, μ_2) in time $t_B(x)$, a function usually bounded by some polynomial $p_B(|x|, 1/\epsilon)$ with respect to μ_{1f}. Likewise, f is computable in time $t_f(x)$ which is usually bounded by some polynomial $p_f(|x|, 1/\epsilon)$ with respect to μ_1. An instance x is computed in the obvious manner as $A(x) = B(f(x))$ in time $t_A(x) = t_f(x) + t_B(f(x))$. Then $t_A(x)$ is usually bounded by the polynomial $p_A(|x|, 1/\epsilon) = p_f(|x|, 2/\epsilon) + p_B(p_f(|x|, 2/\epsilon), 2/\epsilon)$. This can be seen as follows: given $\epsilon > 0$,

$$\begin{aligned}
\Pr_{\mu_1}[t_A(x) > p_A(|x|, 1/\epsilon)] &\leq \Pr_{\mu_1}[[t_f(x) > p_f(|x|, 2/\epsilon)] \vee [t_B(f(x)) > p_B(p_f(|x|, 2/\epsilon), 2/\epsilon)]] \\
&= \Pr_{\mu_1}[t_f(x) > p_f(|x|, 2/\epsilon)] \\
&\quad + \Pr_{\mu_1}[[t_f(x) \leq p_f(|x|, 2/\epsilon)] \wedge [t_B(f(x)) > p_B(p_f(|x|, 2/\epsilon), 2/\epsilon)]] \\
&< \frac{\epsilon}{2} + \Pr_{\mu_1}[t_B(f(x)) > p_B(|f(x)|, 2/\epsilon)] \\
&= \frac{\epsilon}{2} + \Pr_{\mu_{1f}}[t_B(x) > p_B(|x|, 2/\epsilon)] \\
&< \epsilon
\end{aligned}$$

where the second inequality follows from the fact that $|f(x)| \leq t_f(x)$. Thus, t_A is usually bounded by a polynomial as claimed and (D_1, μ_1) is polynomial time on average. ■

Step 2 *Suppose μ_2 dominates μ_1 and that (D, μ_2) is polynomial on average. Then so is (D, μ_1).*

Proof: Let $c \geq 0$ witness that μ_2 dominates μ_1, and suppose A solves D in time $t_A(x)$ which is usually bounded by $p_A(|x|, 1/\epsilon)$ with respect to μ_2. Then $t_A(x)$ is usually bounded by $p_A(|x|, 2|x|^{c+2}/\epsilon)$ with respect to μ_1: given $\epsilon > 0$,

$$\begin{aligned}
\Pr_{\mu_1}\left[t_A(x) > p_A(|x|, 2|x|^{c+2}/\epsilon)\right] &= \sum_{n=1}^{\infty} \Pr_{\mu_1}\left[[|x| = n] \wedge [t_A(x) > p_A(n, 2n^{c+2}/\epsilon)]\right] \\
&\leq \sum_{n=1}^{\infty} n^c \cdot \Pr_{\mu_2}\left[t_A(x) > p_A(|x|, 2n^{c+2}/\epsilon)\right] \\
&< \sum_{n=1}^{\infty} n^c \cdot \frac{\epsilon}{2n^{c+2}} < \epsilon.
\end{aligned}$$

■

Together, Steps 1 and 2 clearly imply Theorem 1.

3 Average-case completeness

Using this notion of reducibility, Levin was able to show that there exists a problem complete for a whole class of problems, i.e., a problem to which every other problem in some class is reducible. Thus, he succeeded in identifying a "hardest" problem in some class which therefore can only be polynomial on average if every other problem in the class is as well.

To prove his main theorem, it was necessary for Levin to make some "niceness" assumptions about the distributions he was working with, namely, that they be polynomially computable. Specifically, a distribution μ is *polynomially computable* if there exists a polynomial-time Turing machine that, on input x, computes $\mu(x)$ as a binary rational number. Note that the Turing machine must compute $\mu(x)$, the probability of choosing any string $y \leq x$. This is a stronger condition than the requirement that the density $\mu'(x)$, the probability of choosing x, be computable. Goldreich [5] shows that this is a *strictly* stronger condition if P$\neq$ #P.

(Strictly speaking, some of the distributions described in this paper take on irrational values. However, all of these distributions can be accomodated by relaxing this definition to require only that the function $c\mu$ be polynomially computable for some constant $c > 0$. This relaxation does not detract from any of the results described in this paper.)

We are now ready to introduce the class of *distributional* NP problems, or DistNP. A distributional problem (D, μ) is in DistNP if D is in NP, and μ is polynomially computable.

Note that, even to show that a problem in DistNP requires more than polynomial time in the worst case (let alone on average) is to show that P $\neq$ NP. Thus, such a result seems unlikely. Nevertheless, it is possible to find a complete problem for DistNP. A distributional problem (D, μ) is *complete for* DistNP if (D, μ) is in DistNP, and every other problem in DistNP can be reduced to (D, μ). Thus, if (D, μ) is polynomial time on average, then so is every problem in DistNP.

A problem complete for DistNP

Levin [15] showed that a tiling problem under a near uniform distribution is complete for DistNP. On close analysis of his proof, Goldreich [5] and Gurevich [9] found that Levin's proof could be

simplified by first showing that a generic bounded halting problem is complete, a problem that can then be reduced to tiling.

In particular, the *Bounded Halting Problem* is the following:

Instance: An encoding M of a nondeterministic Turing machine, a word x, and a number t in unary.

Question: Does the machine encoded by M accept x within t steps?

Distribution: The values of t, $|M|$ and $|x|$ are chosen first with probability proportional to an inverse quadratic. Then M and x are chosen uniformly from all strings of the given length. Thus, $\mu'(x) \propto |M|^{-2} \cdot 2^{-|M|} \cdot |x|^{-2} \cdot 2^{-|x|} \cdot t^{-2}$.

Levin's main result (as interpreted by Goldreich) is the following:

Theorem 2 *The Bounded Halting Problem is complete for* DistNP.

Proof: Goldreich [5] gives a clear and careful proof of this theorem. Here I only try to distill some of the main ideas. Let (BH, μ_{BH}) denote the Bounded Halting Problem when decomposed into the associated predicate BH and distribution μ_{BH}. That (BH, μ_{BH}) is in DistNP is easily verified.

Let (D, μ) be any distributional problem in DistNP. We wish to reduce (D, μ) to (BH, μ_{BH}). We know that D is accepted by some nondeterministic machine M with running time bounded by some polynomial p. The usual (worst-case) reduction of D to BH would map an instance x of D to instance $\langle M, x, p(|x|)\rangle$ of BH. The problem with this reduction is that it fails the domination condition: an extremely common instance with probability, say, $|x|^{-2}$ gets mapped to a far rarer instance with probability proportional to $|x|^{-2} \cdot 2^{-|x|}$.

The main insight needed to overcome this difficulty is the following: We would like to map every instance x to the shortest string possible since μ_{BH} assigns higher probability to shorter instances. Moreover, if x is a very common instance (so that $\mu'(x)$ is large), then x can be more compactly represented using the (polynomially computable) function μ.

In particular, x can be encoded by any fraction in the interval $(\mu(x-1), \mu(x)]$, where $x-1$ is the predecessor of x. Furthermore, such an encoding can be efficiently and uniquely decoded using a kind of binary search since μ is polynomially computable. Finally, note that there always exists a fraction $\alpha_\mu(x)$ in this interval whose binary expansion is of length $\lg(1/\mu'(x)) + O(1)$.

Thus, any string x can be efficiently encoded by $C_\mu(x)$, the shorter of x itself and $\alpha_\mu(x)$. Note that the density on strings induced by this compression scheme is very flat — every string has density $O(2^{-|x|})$. Note also that there exists a Turing machine M_μ that, given a compressed string $C_\mu(x)$, first decodes x, and then (nondeterministically) simulates M on x to decide $D(x)$ in time bounded by some polynomial $p_\mu(|x|)$.

The rest of the reduction is straightforward: an instance x of (D, μ) is mapped to $\left\langle M_\mu, C_\mu(x), 1^{p_\mu(x)}\right\rangle$. It can be checked that this reduction now satisfies the domination condition. ■

Other complete problems

So Bounded Halting is a canonical problem complete for DistNP. With this proved, it is possible to prove the completeness of a handful of other problems to which Bounded Halting can be reduced. For example, a straightforward reduction shows that the following variant of the tiling problem is complete:

Instance: A set of "legal" tiles $L \subseteq \mathcal{R}^4$, each labeled in the corners with one of the twenty-six letters of the Roman alphabet $\mathcal{R}$; a number t in unary; and a legal string σ of tiles from L of length at most t.

Question: Can σ be extended to a tiling of a $t \times t$ square using tiles only from L?

Distribution: L is chosen uniformly at random from $\mathcal{R}^4$, t is chosen with probability proportional to t^{-2}, $|\sigma|$ is chosen uniformly from $\{1, \ldots, t\}$, and σ is chosen uniformly from all legal strings of this length.

In a standard reduction, an instance $\langle M, x, 1^t \rangle$ is mapped to $\langle L_0, \sigma, 1^t \rangle$ where L_0 encodes the legal computations of a universal Turing machine, and σ encodes $\langle M, x \rangle$. Since L_0 has some constant probability of being chosen under the above distribution, the domination condition is satisfied. This is exactly why such a reduction succeeds in this case, but is bound to fail in others.

For example, in the standard proof of the NP-completeness of satisfiability of CNF formulas, the computations of a Turing machine are encoded not in one place (such as the set of legal tiles in the tiling problem), but rather it is encoded again and again throughout the formula. More specifically, if x_{ij} represents the j-th bit of an instantaneous description of the encoded machine M on the i-th step, then x_{ij} is some function of some other variables $x_{(i-1)j'}$ which can be encoded by a constant length formula. However, whatever this formula is, it must be repeated for each variable x_{ij}, and the chance of such a repetition of this pattern occurring in a random formula becomes exponentially small. This appears to be the primary reason why it has proved so difficult to show the completeness of other more natural problems.

Nevertheless, Venkatesan and Levin [17] did manage to come up with a graph coloring problem that is hard on average. Their result is interesting and surprising because graph problems have until now proved to be an excellent source of NP-complete problems that are easy on average. Their technique is also of interest: in essence, they prove their hardness result using the very methods used in the past to prove the easiness of other random graph problems.

Here is a description of the problem they consider: Let G be a directed graph, each of whose edges has been assigned one of the four colors blank, black, red or green. A *spot* is an induced three-node, unlabeled subgraph of G. The coloration of G, denoted $\mathcal{C}(G)$, is the set of all spots of G.

Their random graph coloring problem can be stated as follows:

Instance: A directed (uncolored) graph G, a coloration $\mathcal{C}$, and a number k.

Question: Can the edges of G be colored so that $\mathcal{C}(G) = \mathcal{C}$, and so that the number of blank edges is exactly k?

Distribution: $\mathcal{C}$ is chosen uniformly, $|G|$ is chosen with probability proportional to $|G|^{-2}$, k is chosen uniformly from $\{1, \ldots, |G|\}$, and G is chosen uniformly from all graphs of size $|G|$ (i.e., each edge is present with probability 1/2).

Venkatesan and Levin's main result is a proof that this problem is complete for DistNP. They prove this by a *randomized* reduction from Tiling (or from Bounded Halting). That is, an instance of Tiling is mapped by a randomized function f to one of a number of possible instances of the graph coloring problem.

Here I sketch some of the high-level ideas of their reduction, which I find easier to think about as a direct reduction from Bounded Halting rather than Tiling. Let $\langle M, x, 1^t \rangle$ be a Bounded Halting Problem instance. Such an instance is mapped to a graph G on $\Theta(t^2)$ vertices. This graph is

random, except for the requirement that it have a number of features. The most important of these is an embedded $t \times t$ grid of t^2 vertices; that is, each vertex v_{ij} of this grid is connected to $v_{(i+1)j}$ and $v_{i(j+1)}$. This grid is where the computation of a universal Turing machine is simulated: the coloring of the grid encodes the time-space history of the Turing machine in the usual manner.

The graph G has a number of other features that together with the chosen coloration C_0, ensure the coloring of this grid is in conformity with the computation of a universal Turing machine on $\langle M, x \rangle$, and thus that the graph be colorable if and only if $\langle M, x \rangle$ is accepted. This part of the reduction falls into the standard paradigm used in (worst-case) reductions of building "gadgets" to force a particular behavior. What is new is their construction of a graph with features that are likely to be contained by a large fraction of all graphs. That is, they show that an (entirely) random graph will have all of the required features with probability at least $1/n^c$ for some constant $c > 0$, and thus they are able to show that their reduction satisfies the domination condition.

An incompleteness result

As mentioned above, Venkatesan and Levin's reduction is *randomized*. It is not hard to modify Theorem 1 to show that, if f is a randomized function reducing (D_1, μ_1) to (D_2, μ_2), and (D_2, μ_2) is solved in polynomial time on average by a randomized Turing machine, then so is (D_1, μ_1).

In fact, it turns out that the distributional graph coloring problem described above cannot be proved complete if only deterministic reductions are allowed: an interesting result of Gurevich [6, 9] shows that if μ is "too close" to being uniform, then the distributional problem (D, μ) cannot be complete for DistNP. I close this section with a description of this intriguing result.

A distribution μ is said to be *flat* if there exists a constant $\delta > 0$ such that for all $x \in \Sigma^*$, $\mu'(x) \leq 2^{-|x|^\delta}$. Thus, each instance has very low density. Note that μ_{BH} described above is *not* flat since, by fixing M and x, and allowing t to grow, we can find strings of density proportional to $|\langle M, x, 1^t \rangle|^{-2}$. On the other hand, the distribution on Venkatesan and Levin's graph coloring problem is flat.

Below, Exp (NExp) is the set of decision problems accepted by deterministic (nondeterministic) Turing machines in exponential (i.e., $2^{n^{O(1)}}$) time. The proof of Gurevich's theorem is omitted.

Theorem 3 *Let* $(D, \mu) \in$ DistNP, *and suppose* μ *is flat. Then* (D, μ) *is not complete for* DistNP, *unless* Exp = NExp.

4 Easier than easy on average

In this section, I propose a natural liberalization of the notion of easy on average that seems to have been overlooked in the past.

The standard notion of easy on average described in Section 2 requires that there exist an algorithm for solving the decision problem that is always correct — that is, the algorithm must find a *certificate* that justifies its answer. Thus, for example, it is not enough in deciding graph 3-colorability to observe that most graphs are not 3-colorable — an algorithm must certify that the given graph is not 3-colorable, for instance, by finding a 4-clique.

In some applications, this requirement may be too strong. For example, in designing a pseudo-random bit generator, one would like to say that an adversary is unlikely to guess the next generated bit *by any means*. It is irrelevant in such a setting whether the adversary has a certificate of the value of the bit — only that he can make a reasonable guess.

The formalism for such a liberalization is motivated by the characterization of polynomial on average given by Lemma 1. Recall that this definition states that a Turing machine M solves a

distributional problem (D, μ) in polynomial time on average if there exists a polynomial p such that, for all $\epsilon > 0$,

$$\Pr_\mu [t_M(x) > p(x, 1/\epsilon)] < \epsilon$$

where $t_M(x)$ is M's running time on input x. Note that if M's computation is cut off after $p(x, 1/\epsilon)$ steps (and M is forced to output a default value if it is not yet finished), then the probability that a correct answer is output exceeds $1 - \epsilon$.

This suggests the following definition: A Turing machine M solves distributional problem (D, μ) *approximately in polynomial time* if, for all $\epsilon > 0$,

$$\Pr_\mu [M(x, \epsilon) \neq D(x)] < \epsilon$$

where the probability is over random choices of x (according to μ). Furthermore, M's running time must be polynomial in $|x|$ and $1/\epsilon$.

Note that Theorem 1 can easily be modified to handle reductions in which f is only approximately computable. Then if f reduces (D_1, μ_1) to (D_2, μ_2), and (D_2, μ_2) is approximately solvable in polynomial time, then so is (D_1, μ_1). In particular, this shows that the Bounded Halting Problem, as well as the other problems described in Section 3, are complete under such approximate reductions. Thus, if the Bounded Halting Problem is approximately solvable in polynomial time, then so is every other problem in DistNP.

Let AverP denote the class of distributional problems (D, μ) for which μ is polynomially computable and which are solvable in polynomial time on average. (This definition differs slightly from those given by Goldreich [5] and Ben-David et al. [2]) Let ApproxP be the class of distributional problems (D, μ) for which μ is polynomially computable, and which are approximately solvable in polynomial time. From the preceding remarks, we have:

Theorem 4 AverP $\subseteq$ ApproxP.

Containment in the opposite direction is apparently an open question, though my guess is that AverP is properly contained in ApproxP. As suggestive evidence (but not proof), I would cite various problems which are approximately solvable, but for which the existence of an algorithm running in time polynomial on average seems uncertain. These are described in the following subsections.

ApproxP and AverP algorithms for finding cliques

To start with, consider a variant of the clique problem. The general clique problem on random graphs (i.e., graphs in which each edge is independently present with probability 1/2) is known to be solvable by an algorithm with expected running time $n^{O(\log n)}$ [9]. Solving Clique in polynomial time on average is open, although some progress was made by Phan Dinh, Le Cong, and Le Tuan [16] in this regard by restricting the edge probabilities or the total number of edges in the graph. Below, I have obtained positive results by instead restricting the size of the clique being sought.

Let Clique($k(n)$) be the problem of deciding if an n-node graph has a $k(n)$-clique. Let μ_0 be a standard uniform distribution on graphs, i.e., the number of vertices n is chosen with probability proportional to n^{-2}, and each edge is present with probability 1/2. Then for certain choices of $k(n)$, (Clique($k(n)$), μ_0) is in ApproxP, as proved below. Note that the "standard" proof that the Clique problem is NP-complete as described by Hopcroft and Ullman [12] asks only whether the given n-node graph contains an $(n/3)$-clique, and thus shows that Clique($n/3$) is NP-complete.

Theorem 5 *Assume $k(n) = \omega(\log n)$ is polynomially computable. Then* (Clique($k(n)$), μ_0) $\in$ ApproxP.

Proof: The algorithm A that approximately solves this problem is very simple: On input ϵ and an n-node graph G, A compares ϵ with

$$\binom{n}{k} \cdot 2^{-\binom{k}{2}}$$

where $k = k(n)$. If ϵ is larger than this number, then A just says "no" (the Nancy Reagan heuristic), since this number bounds the probability that a random n-node graph contains a k-clique. Otherwise, if ϵ is very small, A does a brute-force search of all $\binom{n}{k}$ subsets of k vertices to determine if G has a k-clique. Since ϵ is so small in this case, and since $k(n) = \omega(\log n)$, the running time is only polynomial in n and $1/\epsilon$. ■

It seems unclear in general how to find a certificate that the graph does not contain a $k(n)$-clique in the first case above that ϵ is large. On the other hand, when $k(n) = cn$ for some constant $0 < c < 1$, I was able to devise a polynomial time on average algorithm:

Theorem 6 *Let $0 < c < 1$ be fixed. Then* $(\text{Clique}(cn), \mu_0) \in \text{AverP}$.

Proof: The algorithm A for solving this problem in polynomial time on average is a bit more complicated than that in the previous theorem. Given a graph $G = (V, E)$ on n vertices, the algorithm works as follows:

1. Let $d = 1 + \lceil \lg(1/c) \rceil$. For each set S of d vertices, compute the number of vertices that are common neighbors of all the vertices of S. (A vertex is its own neighbor, and is also the neighbor of every other vertex with which it shares an edge.) If for every such set S this number is less than cn, answer "no."

2. Otherwise, do the same thing for every set of $\ell = 3\lceil \lg n \rceil$ vertices. Again, if the number of common neighbors of every set of ℓ vertices is less than cn, answer "no."

3. Otherwise, do a brute-force search to determine if the graph contains a cn-clique.

Note that in cases 1 and 2, a certificate is obtained that the given graph has no cn-clique (assuming n is so large that cn exceeds d and ℓ) since, if the graph did contain a cn-clique, then any subset of the nodes forming that clique would have at least cn common neighbors.

Let S be a fixed set of d vertices. Let N be the random variable describing the set of vertices in $V - S$ that are adjacent to every node of S when G is randomly chosen. Then the probability that a vertex $v \in V - S$ is in N is easily computed to be 2^{-d}. Moreover, this event is independent of other vertices appearing or not appearing in N. Therefore, the cardinality of N is distributed as the number of successes in $n - d$ trials of a Bernoulli variable which succeeds on each trial with probability 2^{-d}. Thus, using Chernoff bounds [1, 11], it can be shown that $|N| \geq cn - d$ with probability at most $2^{-\Theta(n)}$. Note that this also bounds the probability that S has cn common neighbors in V. Therefore, the chance that any set of d nodes has cn common neighbors is at most $\binom{n}{d} \cdot 2^{-\Theta(n)} \leq 2^{-\Theta(n)}$. Note also that step 1 takes time $n^{O(1)}$.

The analysis at step 2 is similar, although Chernoff bounds are unnecessary. The chance that a random graph contains a fixed set of ℓ nodes having $cn - \ell$ common neighbors (again, excluding themselves) is at most

$$\binom{n}{\ell} \cdot \binom{n-\ell}{cn-\ell} \cdot 2^{-\ell(cn-\ell)} \leq n^{\ell} \cdot (n-\ell)^{cn-\ell} \cdot \left(\frac{1}{n^3}\right)^{cn-\ell} \leq n^{3\ell-2cn}.$$

Also, this step takes time $n^{O(\log n)}$.

The final step takes time $n^{cn+O(1)}$. Combining these facts, it follows that the expected running time for a random n-node graph is at most

$$n^{O(1)} + 2^{-\Theta(n)} \cdot n^{O(\log n)} + n^{3\ell-2cn} \cdot n^{cn+O(1)} \leq n^{O(1)}$$

and therefore, by the remarks in Section 2, the algorithm runs in polynomial time on average. ■

An ApproxP algorithm for graph coloring

As mentioned in the introduction, there exist simple-minded algorithms for 3-coloring a graph in polynomial time on average [3, 18]. These algorithms are easily modified for c-coloring, for any constant c. It is apparently open whether $k(n)$-coloring is easy on average when $k(n) = \omega(1)$.

However, it is possible to construct an algorithm, similar to the one in Theorem 5, that approximately $k(n)$-colors a random graph when $k(n) = o(n/\log n)$. Below Color($k(n)$) is the problem of deciding whether a graph is $k(n)$-colorable.

Theorem 7 *Assume $k(n) = o(n/\log n)$ is polynomially computable. Then* $(\text{Color}(k(n)), \mu_0) \in$ ApproxP.

Proof: The algorithm A that approximately solves this problem is very similar to the one desribed in the proof of Theorem 5. Given $\epsilon > 0$ and an n-node graph, A compares ϵ with some threshold value. If ϵ is larger than this value, then A answers "no;" otherwise, a brute-force search ensues.

An appropriate threshold value is

$$\theta = n! \cdot n^k \cdot 2^{n/2-n^2/2k}$$

where $k = k(n)$. First, if ϵ is less than θ, then all k^n possible colorings of the graph can be tried in time polynomial in n and $1/\epsilon$ since $k(n) = o(n/\log n)$. If ϵ is more than θ, then a simple "no" suffices since θ bounds the probability that a random n-node graph is k-colorable. To see that this is so, note that a graph G is k-colorable if and only if its vertex set V can be partitioned into k independent subsets. (A set is *independent* if no two vertices in the set are connected.) Thus, the probability that G is k-colorable is at most

$$\sum 2^{-\sum_{i=1}^{k} \binom{|A_i|}{2}}$$

where the sum is over all partitions of V into k nonempty blocks $A_1, \ldots, A_k$. The number of such partitions is loosely bounded by $n! \cdot n^k$. Furthermore,

$$-\sum_{i=1}^{k} \binom{|A_i|}{2} = \frac{n}{2} - \sum_{i=1}^{k} \frac{|A_i|^2}{2} \leq \frac{n}{2} - \frac{n^2}{2k}$$

by a convexity argument. It follows that θ bounds the probabiiity of a random graph being k-colorable. ■

5 Comparing complexity classes

Levin's paper opened the way to the study of a whole family of new complexity classes. This section explores some of the relationships among these classes.

We have already discussed DistNP, AverP and ApproxP. A fourth class discussed by Goldreich [5] and attributed to Ronnie Roth is the class AverNP, a natural liberalization of DistNP.

Specifically, AverNP consists of those problems (D, μ) for which μ is polynomially computable and some nondeterministic machine M solves D in time polynomial on average. That is, there exists a function $\ell : \Sigma^* \to \mathbf{N}$ such that $1^{\ell(x)}$ is computable in polynomial time on average, and there exists a computation of M that accepts x in $\ell(x)$ steps if and only if $D(x) = 1$.

Goldreich makes the interesting observation that every problem in AverNP is reducible to the Bounded Halting Problem by a simple modification of the proof of completeness for DistNP. Note that this does *not* imply that AverNP $\subseteq$ DistNP due to the fact that the reduction used may require more than polynomial time in the worst case. On the other hand, it is easy to see that AverNP contains both AverP and DistNP.

The purpose of Section 3 was to find a problem in DistNP that is not in AverP unless every other language in DistNP is as well, i.e., unless DistNP $\subseteq$ AverP. This last assumption in fact can be reduced to a more comfortable assumption from the theory of worst-case complexity. This is proved by the next theorem which is a slight generalization of one proved by Ben-David et al. [2]:

Theorem 8 *If* $\mathrm{DTime}(2^{O(n)}) \neq \mathrm{NTime}(2^{O(n)})$ *then* $\mathrm{DistNP} \not\subseteq \mathrm{ApproxP}$.

Proof: Suppose to the contrary that DistNP $\subseteq$ ApproxP. Let D be a decision problem in $\mathrm{NTime}(2^{O(n)})$. Then the unary problem $D'(1^x) = D(x)$ is in NP. (Here, string x is associated with a natural number in the usual way.) Consider the distribution $\mu'(1^x) = z/x^2$, where z is some normalization constant. Then μ is polynomially computable, and so $(D', \mu) \in \mathrm{DistNP} \subseteq \mathrm{ApproxP}$. Thus, there exists a Turing machine M for which

$$\Pr_\mu [M(y, \epsilon) \neq D'(y)] < \epsilon$$

and that runs in time polynomial in $|y|$ and $1/\epsilon$. Note that this condition implies that if $\epsilon = \mu'(1^x) = z/x^2$ then $M(1^x, \epsilon) = D'(1^x) = D(x)$. Therefore, on input x, $M(1^x, z/x^2)$ computes $D(x)$ in time polynomial in $x = \Theta(2^{|x|})$. ■

A fundamental question concerning these average-case complexity classes concerns their relationship to other worst-case complexity classes. For example, if $(D, \mu) \in \mathrm{AverP}$, what can be said about the complexity of D? The answer is: very little. As an extreme example, if μ concentrates all its probability mass on a single point, then there obviously exists a very fast (constant time on average) algorithm for (D, μ), despite the fact that there exist languages that require an arbitrarily great amount of time to decide.

A more reasonable question then is to ask about the complexity of D restricted to the support set of μ. Specifically, let $D|_\mu$ be the decision problem defined by $D|_\mu(x) = D(x)$ if $\mu'(x) > 0$, and $D|_\mu(x) = 0$ otherwise. Further, for distributional complexity class $\mathcal{C}$, let $\overline{\mathcal{C}}$ denote the class of decision problems

$$\overline{\mathcal{C}} = \{D|_\mu : (D, \mu) \in \mathcal{C}\}.$$

Now we can re-ask our question: How does $\overline{\mathrm{AverP}}$ fit into the time complexity hierarchy?

The following theorem answers this question more generally:

Theorem 9

- $\overline{\mathrm{ApproxP}} \subseteq \mathrm{Exp}$, *and*
- $\overline{\mathrm{AverNP}} \subseteq \mathrm{NExp}$.

Proof: I prove the first part only; the second part is similar.

Let $(D, \mu) \in$ ApproxP. Then there exists a Turing machine M that solves (D, μ) approximately so that

$$\Pr_\mu [M(x, \epsilon) \neq D(x)] < \epsilon$$

and that runs in time polynomial in $|x|$ and $1/\epsilon$. Since μ is polynomially computable, we can easily decide whether $\mu(x) = 0$. Moreover, since the length of the output of a machine computing μ is bounded by the machine's running time; it follows that, for some polynomial p, $\mu'(x) \geq 2^{-p(|x|)}$ if $\mu'(x) > 0$. Since, as noted in the preceding theorem, $M(x, \mu'(x)) = D(x)$, it follows that $D|_\mu(x)$ can be decided in exponential time. ■

Finally, it can be shown that this last theorem is the best that can be proved:

Theorem 10

- Exp $\subseteq \overline{\text{AverP}}$, *and*
- NExp $\subseteq \overline{\text{AverNP}}$.

Proof: Again, I only prove the first part.

Let $D \in$ Exp. Then D is accepted by some machine M in time $2^{p(n)}$ for some polynomial p. Let $\mu'(x) \propto 2^{-(p(|x|)+2|x|)}$. Then μ is polynomially computable, and $D|_\mu = D$. Moreover, M accepts D in polynomial time on average (with respect to μ) since, by an easy computation,

$$\sum_{x \in \Sigma^*} \mu'(x) \frac{t_M(x)}{|x|} \leq \sum_{x \in \Sigma^*} \mu'(x) \cdot 2^{p(|x|)} < \infty.$$

■

Together, these theorems completely characterize $\overline{\text{AverP}}$, $\overline{\text{ApproxP}}$ and $\overline{\text{AverNP}}$.

Corollary 1

- Exp $= \overline{\text{AverP}} = \overline{\text{ApproxP}}$,
- NExp $= \overline{\text{AverNP}}$, *and*
- NP $= \overline{\text{DistNP}}$.

Further, we are now ready to fully characterize (almost) the containment relationships among these average-case complexity classes. This is summarized in the containment graph in Figure 1. An edge directed from A to B indicates that class A is contained in class B. A dashed edge indicates that the containment question is open. This graph assumes that NP $\neq$ Exp, and DTime$(2^{O(n)}) \neq$ NTime$(2^{O(n)})$.

Note that there remain two unresolved containment questions. Namely, is ApproxP contained in either AverP or AverNP?

6 Summary

In this paper, I have reviewed much of what is known about average-case complexity, though I certainly have not covered everything. I have described Levin's framework for studying average-case complexity, and have discussed some of the few known complete distributional problems. I have also suggested a more relaxed notion of "easy on average," which captures the notion of a problem that can be solved "approximately." Finally, I have discussed how the new average-case complexity classes relate to one another.

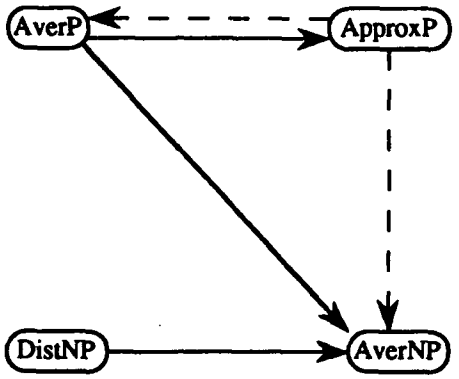

Figure 1: The containment graph for some average-case complexity classes

Acknowledgements

This is a revised version of a paper prepared as part of my "area exam." Thanks first to Silvio Micali, Charles Leiserson and Michael Sipser for serving on my exam committee, and for their comments and advice. Thanks also to Rafail Ostrovsky and Joel Wein for their help and encouragement.

References

[1] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18(2):155–193, April 1979.

[2] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 204–216, May 1989.

[3] Edward A. Bender and Herbert S.Wilf. A theoretical analysis of backtracking in the graph coloring problem. *Journal of Algorithms*, 6(2):275–282, June 1985.

[4] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.

[5] Oded Goldreich. Towards a theory of average case complexity (a survey). Technical Report 531, Technion Computer Science Department, December 1988.

[6] Yuri Gurevich. Complete and incomplete randomized NP problems. In *Proceeding of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pages 111–117, October 1987.

[7] Yuri Gurevich. The challenger-solver game: Variations on the theme of P=?NP. *Bulletin of the European Association for Theoretical Computer Science*, October 1989.

[8] Yuri Gurevich. Matrix correspondence problem is complete for the average case. Unpublished manuscript, November 1989.

[9] Yuri Gurevich. Average case completeness. *Journal of Computer and System Sciences*, To appear.

[10] Yuri Gurevich and David McCauley. Average case complete problems. Unpublished manuscript, April 1987.

[11] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[12] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[13] David S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 5(2):284–299, June 1984.

[14] Leonid A. Levin. Problems, complete in "average" instance. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, page 465, April 1984.

[15] Leonid A. Levin. Average case complete problems. *SIAM Journal of Computing*, 15(1):285–286, February 1986.

[16] Phan Dinh Dieu, Le Cong Thanh, and Le Tuan Hoa. Average polynomial time complexity of some NP-complete problems. *Theoretical Computer Science*, 46(2, 3):219–327, 1986.

[17] Ramarathnam Venkatesan and Leonid A. Levin. Random instances of a graph coloring problem are hard. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 217–222, May 1988.

[18] Herbert S. Wilf. Backtrack: An $O(1)$ expected time graph coloring algorithm. *Information Processing Letters*, 18(3):119–121, March 1984.

OFFICIAL DISTRIBUTION LIST

DIRECTOR Information Processing Techniques Office Defense Advanced Research Projects Agency (DARPA) 1400 Wilson Boulevard Arlington, VA 22209	2 copies
OFFICE OF NAVAL RESEARCH 800 North Quincy Street Arlington, VA 22217 Attn: Dr. Gary Koop, Code 433	2 copies
DIRECTOR, CODE 2627 Naval Research Laboratory Washington, DC 20375	6 copies
DEFENSE TECHNICAL INFORMATION CENTER Cameron Station Alexandria, VA 22314	12 copies
NATIONAL SCIENCE FOUNDATION Office of Computing Activities 1800 G. Street, N.W. Washington, DC 20550 Attn: Program Director	2 copies
HEAD, CODE 38 Research Department Naval Weapons Center China Lake, CA 93555	1 copy